

offering an accredited Bachelor program for Media and Computing since the 2001/02 school year with over 460 students enrolled at six participating institutions. A Master program began in the 2004/05 school year.

In the curriculum of the Bachelor program for Media and Computing there is a 5 ECTS [1] credit Software Engineering module in the third semester. The purpose of this course is to give the students an introduction to the basics of traditional Software Engineering. A second software-engineering module is offered in the Master program that offers in-depth discussion of current topics such as Model-Driven Architectures, and there is an advanced Java programming module in the third semester of the Bachelor program, parallel to the Software Engineering module, that discusses object-oriented concepts and design patterns.

2. Course Design

2.1 Design Objectives

One of the guiding concepts of the VFH is that internet-based learning should be essentially different from just reading a linear textbook presented in PDF format on the web. Reading such documents online or printing them on moderate quality home printers will often be inferior to working with a traditionally published book (10). Rather, in producing e-learning material, the learning situation needs to be improved by incorporating "e"-specific features such as non-linear exposition and online interaction, communication, and cooperation into the learning experience (12)(18).

In the VFH, students of a variety of vocational backgrounds and in various personal settings study their material individually, but interaction with each other and their course mentor during the term is encouraged and in many courses required. Both aspects, the production of course materials adequate for individualized learning, and the organization and incorporation of interaction over the internet, are given particular attention in course design.

2.2 The VFH Hypermedial Didactical Structure

Multimedia teaching materials must be presented in a screen-adapted layout and format. There must be a clear navigational structure and a recognizable set of visualization types used in the module. This may sound simple, but it is a challenge both to designers and to producers of the materials. In addition, the "internet paradigm" of nonlinear access and free browsing should not only be enabled, but rather supported by adequate indexing and cross-referencing, in order to allow for individualized learning paths.

The VFH team in Berlin together with the VFH ergonomics expert team [2] has developed a hypermedial didactical structure given in Figure 1 that has shown itself to be very effective for our purposes (see (4) for more detail). A workflow was also developed to facilitate the production of the materials.

A VFH module consists of a set of learning units, which form a didactical sequence, but can be read in any order. The module is accessed through a so-called shell. Besides the basic navigation, the shell offers a hyperlinked table of contents, a hyperlinked glossary, links to pages with chapter descriptions, indexes of pictures, animations and exercises, a download link for an off-line version of the text, literature lists, optional additional subjects and sources, a copyright page, etc.

Every learning unit begins as usual for distance learning materials with an introduction page that gives the learning goals for this unit and some information about the time expected to be invested in the course. Then a series of chapters follow, which themselves have subchapter sequences. No deeper structure is used, to keep learners from getting lost while reading. Since three- and often four-level structures are very common in German, this meant rearranging some of the material and "promoting" sub subchapters, introducing a new chapter structure.

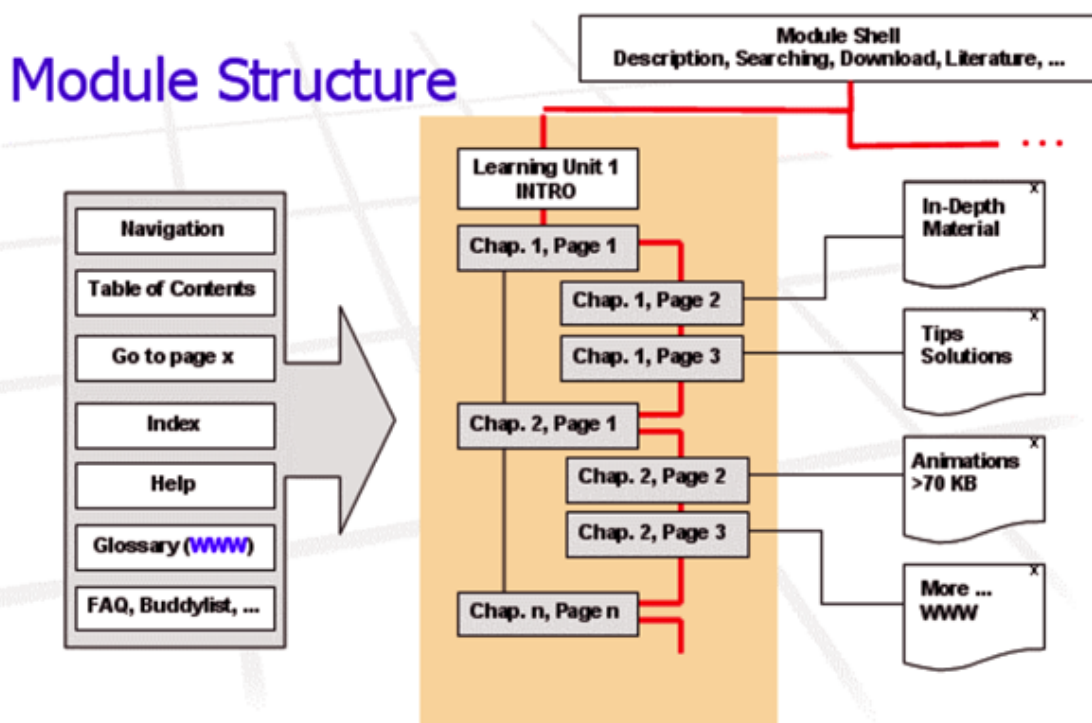



Figure 1 - Structure of a Learning Unit

Supplementary materials are offered on extra pages that open up in separate browser windows, so as not to destroy the reading context. Links to material on the Internet are specially marked with an icon () and also open new windows. Other kinds of supplementary material such as learning tips or animations (which are kept to a minimum size if at all possible to minimize download times) are also handled in this manner.

A common navigation between chapters, subchapters and pages is provided, with pages numbered sequentially through the learning unit. The total number of pages (e.g. 7/31) is given so that the learner is aware of how many pages must still be covered. In order to ease using the material for learning there are also links to a glossary, a linked table of contents, the course shell, and an index on every page.

The glossary and the index need to be given extra attention, as they form the keys to non-linear access to the teaching material, one of the essential features of online modules. With the myriad terminology used in Software Engineering, sometimes with multiple or contradictory usage, a glossary cross-linked to the teaching units is indispensable for working with the material and referencing other literature with differing terminology.

Visualizations are also vital for understanding the often abstract material which is offered. In contrast to textbooks, which are restricted to static graphics, e-learning modules can use animated and interactive graphics for visualizing purposes. For structural clarity, we do not mix these techniques, but have three basic patterns: static graphics, animations for time-based developments, and roll-overs to present master-detail relationships. Often, students print out parts of the material for making annotations. Therefore, it is desirable to provide a static, printable version with every dynamic visualization. Figure 2 is such a static representation of an animated visualization for stepwise refinement. In the on-line version of this article clicking on the picture will open a window with the animation itself.

Within text, animation has also proven useful for self-test elements. We have multiple choice, true/false, matching and fill-in-the-blank exercises done in Flash that can be used for self-test exercises. However, these need to be carefully designed so that they do not suggest to a student that they have understood material when in fact they just can repeat facts. These animations also need static representations for printing.

What is still missing from this implementation is the possibility of entering annotations for the module online. Many students thus print out pages from the screen (which are then, of course, not easy to read as they are optimized for screen reading) in order to mark them up. With the modules stored in XML format, it would be easy to produce printable PDF versions on demand. Another approach would be to allow online annotation by embedding the HTML in a WikiWiki [3]. As resources permit, we will be experimenting with both solutions.

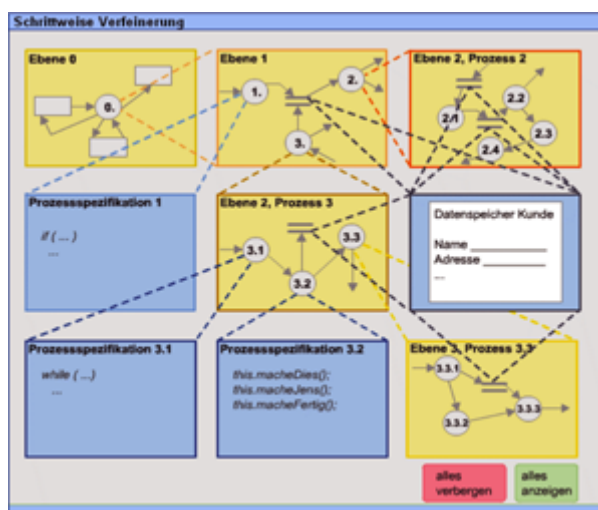


Figure 2 - Stepwise Refinement Animation (Flash)

With the shell as a toolbox for accessing the learning material, we find that the VFH hypermedial structure is particularly well suited for online learning and allows the learner to individualize his or her learning path in a very flexible way.

2.3 Individualization through Learning Objects?

In e-learning, a much discussed approach is the production of "Learning Objects" (LOBs), representing atomized learning units designed for online presentation. They are then marked up with metadata, SCORM and LOM being the prevalent standards, with the aim of freely combining and reusing "learning objects" in setting up learning contexts (e.g. courses). The eventual goal is to be able to automatically construct an individualized e-learning course according to the learner's knowledge and needs.

At present it is not possible to construct any software engineering course from LOBs, because there are practically none. Even the MuSoft project [4], also funded by the German BMBF, only has less than 100 objects listed on their web pages in August of 2005, ranging from videos over slides to complete learning units. There was nothing at all publicly available when we began our work. We could have started filling this gap by constructing our course as a set of LOBs and mark them up according to one of the standards. But we share other authors' disbelief in the didactical adequacy of the LOB approach (7)(13).

Let us, in order to explain our concerns, assume that we would turn subchapters into LOBs. It is not at all clear at what level of detail to form an LOB, but subchapters are a likely unit of omission and replacement in an individualized course, so the choice seems reasonable. The following problems occur immediately:

1. Terminology:

As in all non-exact sciences, learning software engineering consists to a large extent of learning and applying terminology to a given subject matter. Unfortunately, there is not one universally accepted, complete software terminology. Hence each subchapter has to rely on the terminology introduced and explained in previous chapters. It will probably be incomprehensible without its context. A glossary would clearly help to bridge some of the gaps, but it cannot be sufficient for terminology with deeper methodological connotation.

2. Interdependence:

Terminology is just one aspect that makes units of a didactically well designed course highly interdependent. If the markup would allow specifying references to other LOBs, we would find that every LOB has backward references to very many others. From an object-oriented point of view one could think of chapter "facades" to simplify the reference structure. But this type of information encapsulation is the exact opposite of the didactical one in which you use back references to the very details of a concept in order to strengthen and refine the learner's command of it. Griffiths and Garcia comment in Koper (6) that "[the object-oriented] analogy, which runs in perfect accord with the conduit metaphor, runs contrary to much of what we know we know how human beings construct meanings from language, texts, images, etc. ..."

3. Objective individualization:

Of course, every learner should have an individualized way of studying a course, adapted to his or her knowledge and needs. But there is no objective measure of a learner's knowledge, covering soft factors such as the relative difficulty of accessing the knowledge, its initial context, or the learner's attitude towards the content, which strongly influence the availability of a certain knowledge in a learning context. Crutzen points out in (2) that "Learners are not objects with a predictable behavior

engaged in a learning process that is reduced to formal and planned acting." Hence, course individualization should not be attempted in an "objective" manner, modeling the learner as an object with knowledge and requirement attributes. It should rather respect the learner as an individual with differently shaded knowledge and emotions like interest, fear or anxiety, who is best suited to decide on his or her optimal learning path, including chapters that repeat forgotten material, and skipping parts of limited relevance to his or her particular focus.

Therefore we believe in e-learning courses that support individualization by the individuals, through

- giving the learner the security of having access to the full set of learning units
- supporting non-linear access to the material as much as possible, using e.g. chapter abstracts, time requirements, self-tests, and cross-linked content tables, indexes and glossaries
- clearly stating the requirements for passing the course.

We find that using the VFH hypermedial didactical structure, we meet these criteria, whereas using LOBs we don't.

2.4 Didactical Outline of the Course

Before beginning the technical production, a didactical outline for the course had to be set up. In all distance teaching situations, it is vital to give the students an idea of how much they need to achieve every week, in order to prevent them from starting enthusiastically, but then yielding to pressure from other areas of their lives and postponing most of the work to the end of the semester. Hence, the basic course materials were first divided up into 16 chapters of similar length, roughly corresponding to the typical 16 week term of a German college.

In order to synchronize the progress of the group, which is essential for team cooperation, each chapter ends with a set of questions or small exercises which have to be completed in at the end of a pre-determined week. The results and questions arising from the text are then to be discussed in a weekly chat.

A second type of exercises requires the students to collectively accumulate information e.g. through web research, or cover portions of a complex exercise which then have to be integrated. The results are to be posted on the learning management system's bulletin board and will be discussed and combined in a subsequent chat. This way, the students are introduced to the use of asynchronous communication and learn to profit from their classmates' work - both basic skills for team cooperation.

Finally, there are two exercises to be done in face-to-face classroom sessions, because our experience is that personal encounter raises the effectivity and satisfaction of subsequent online communication substantially. The classroom sessions are set aside for larger modeling exercises in teams, because this is where constant feedback from the mentor is most needed.

2.5 Content Adaptation

With no adequate learning objects to be found, and a very limited budget of only 75.000€, the only feasible way of producing an online learning module for software engineering was to start with an existing textbook and reshape it according to the VFH hypermedial didactical structure. This would comprise flattening and balancing the chapter hierarchy, presenting the given text in the specified screen-reading format, putting it into a shell structure and adding the required referencing, navigation, planning and assessment tools, enriching the text with multimedia visualizations, and adding interaction and cooperative elements.

It proved unexpectedly difficult to find course materials to start with, because for several reasons, authors were reluctant to offer their courses for an online presentation. We were fortunate to be able to convince M. Stanierowski to permit us to use her text on Software Engineering for a Business and Computing program used at the distance learning university of applied sciences in Hamburg, the Hamburger Fern-Hochschule [5].

With the initial problems of finding the basic material, the teaching module had to be produced in only nine months time, with a three months overlap between course production and teaching usage. We started with a team of students doing this as a project, with a full-time producer added to the team just for the last five months. The successful completion is owed, we believe, to the well specified hypermedial target structure and the fact that Stanierowski's course material was evenly segmented, due to its nature as a distance education course, and thus needed no rewriting.

3. Structuring the Online Group Process

Adequate online presentation of the teaching materials was an important aspect in creating the e-learning module, but it is not specific to Software Engineering instruction. Rather, teamwork and communication training are specific and vital requirements of a Software Engineering course. These skills are hard to learn in traditional instructional situations (19) and even harder in distance learning scenarios. It is difficult enough to communicate with each other in a face-to-face situation, e.g. for teamwork coordination. Learning how to manage teamwork with internet-based communication requires intensive training.

While it is very powerful and satisfactory when it works, online communication proves often tedious or even frustrating because of setup and synchronization problems. It is unrealistic to make the problems and exercises so hard that the average student would not find it easier to solve them individually rather than working on them through internet communication. So the course needs a didactic concept that enforces online communication in order to train it, and must provide a teaching environment with advanced online communication facilities. After some time of enforced training, students tend to turn to online communication, even where it is not explicitly required.

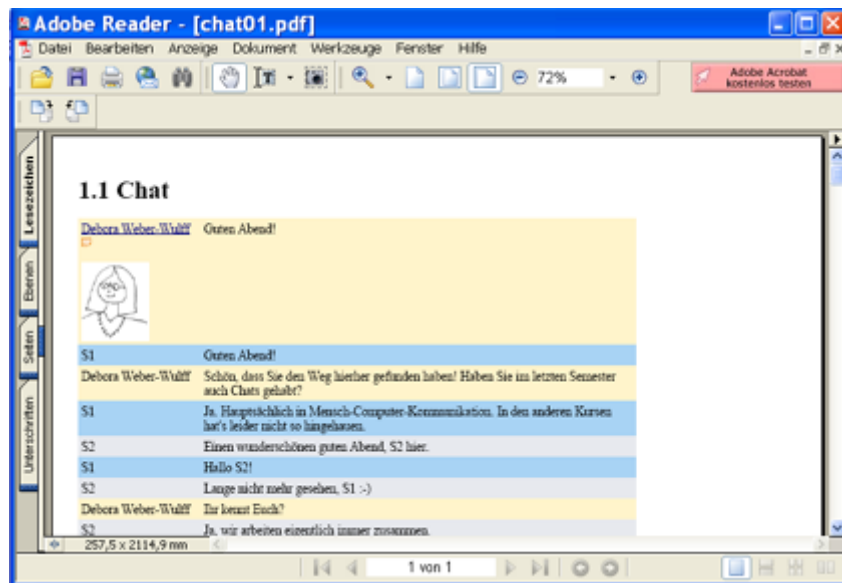
3.1 Asynchronous Communication: Posting Results

In order to properly discuss the teaching material, the learning progress must be synchronized. We do this with mandatory exercises, called "enforced milestones", that are to be handed in, or rather posted on a bulletin board, according to the course schedule. Student comments on others' results are encouraged. The exercises are graded not on content, but on timeliness, accounting for a total of 20% of the grade. Each exercise is worth two points if handed in on time, one point if handed in at all. One can still pass the course by not participating, but having 20% of the final grade already completed upon taking the exam is a strong enticement for people to actually do the work - and since they have already answered the questions, they tend to participate in the chat.

By using bulletin board publishing for handing in the exercises instead of sending an email to the teacher, we create an awareness of common progress in the group. Disclosing one's results rather than hiding them is also a basic concept in web communication, which is increasingly being used in professional software development. Since we do not grade on content, we don't have to worry about plagiarism, and we discuss the results in the chat. As all of the exercises are visible for all of the students, students work extra hard to hand in good results. They also learn the realities of heterogeneous systems from each other - if the teacher can't open their document, they think the teacher is stupid. But if a fellow student can't open their document because it is in some Windows-based format that is not presentable on a Mac or with Linux, then they begin to see the advantages of platform-independent formats.

3.2 Synchronous Collaboration: Chat and Application Sharing

The results of asynchronous collaboration are evaluated in a chat session. As all results are accessible for all participants, a text based chat will suffice in many cases - although with the availability of voice and multimedia "chats", we hardly use text chat anymore. It is important that chats are recorded for later reference. We encourage students to take turns in editing text chats, removing side-remarks and bad misspelling, condensing atomized contributions and anonymizing students; an example of such an edited chat is given in screenshot 1. The edited chat is available in German at (16).

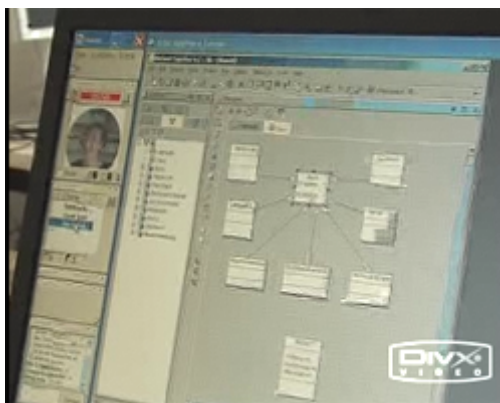


Screenshot 1 - Edited Chat Recording (.pdf Listing)

It is particularly with the modeling exercises that the students need more effective means of synchronous communication than text chat. Trying to interact and discuss a model using only a text based chat is an exercise in frustration. We need to be able to point and to draw in other people's models in order to have effective modeling discussions. Tools are beginning to emerge that permit more than two persons at a time to be active in a diagram [6], but not all of them are easy to use.

What is required is a synchronous communication tool that allows us to interactively use another tool to draw or edit a model. Such a communication tool (similar to application sharing) allows the mentor to demonstrate things, students to ask specific questions by showing the problem, and a group to cooperate on a modeling task. This type of application sharing is vital for any substantial team development. While commenting via text chat would work, it is highly recommendable to have voice chat as an efficient exchange mode accompanying it. Once again, recording of sessions is essential, but editing multimedia sessions is too tedious to be of practical use.

As an illustration, we have filmed a staged replay of a discussion session using the Netucate setup. In the context of discussing an object model for a library information system, there are two student models to be contrasted, one overly simple and one overly complex. The mentor is leading the discussion about generalization and the selection of attributes and the students work together fixing the overly complex model. The film is in German and available for a number of different formats [7].



Screenshot 2 - A Model Discussion using a Collaboration Tool (Demonstration)

Finally, it is worth mentioning that for modeling discussions, the synchronous communication, however powerful, should not replace asynchronous communication: There is a difference between quickly scanning a model draft during a chat discussion and thoroughly reviewing it when it is posted in a bulletin board and the student has to comment on it in writing. Both skills need to be trained. This also serves in training the students in as many online communication techniques as possible, which we consider important, as they are increasingly becoming part of the software professional's working situation.

3.3 Face-to-face interaction

The Software Engineering module starts with a "Telepresence", a virtual meeting of all students at all schools offering the course. Video conferencing was planned, but chat is used because of technical problems. The subject is to start "thinking software" by doing a collective data analysis and requirements elicitation for a CD-dealership, but the idea behind it is also to give students an awareness of who else is registered for the course.

For each school, there are two six-hour weekend meetings for face-to-face instruction. These meetings are compulsory and render 30% of the total credit. The students are generally very willing to participate, because of the effectiveness and the pleasure of working face-to-face. In Software Engineering, we use two weekend meetings for team modeling exercises. Modeling is chosen for several reasons: It is highly interactive, requiring both intensive and quick team interaction and frequent mentor support, which can become tedious using Internet communication; also we can let the students work with different CASE tools without having to deal with licensing and installation topics. In subsequent virtual meetings, CASE tools are frequently used interactively through application sharing, without having to have them installed on every student's computer.

Finally, we have not found a satisfactory internet-based method for conducting assessments that lead to degrees, as we must verify the identity of the candidate. So the final exam, worth 50% of the total credit for this module, is taken in a face-to-face-setting, as well.

3.4 Course Individualization versus Group Process

Having discussed the means and requirements for setting up a group process within a learning community, we have to reconsider the question of individualization, as proposed in the learning objects approach. It is quite obvious that an individualized course set up from learning objects contradicts the idea of group learning. As Crutzen criticizes, "social group processes cannot be represented by an OO-notation, but they are vital for the pedagogical process" (2). It seems as if the individualization concept behind the learning object approach relies on a traditional distant learning paradigm, rather than embracing the communication means that make the Internet such a unique learning setting.

4. Practical Experiences

Having explained the setup and didactics of the online software engineering module, we want to share some of the experiences gained in the last few semesters, especially looking at the mentor's role in the process.

4.1 Material Production and Maintenance

It might seem that it would be quite beneficial to have online materials for teaching tools, as they seem to be easy to adapt to new interfaces or to enrich with new materials. But as soon as the materials are being used by more than one teacher, the versioning and editing problems that are prevalent in print-based instructional materials crop up, along with many additional on-line problems. The new browsers are not backwards compatible, the new material will not work with an old plug-in, there are many copies of the materials that litter the servers, people have taken private copies of pages and have to be informed of the changes made or they will be surprised to find something they wanted to use missing or changed, etc. It is a challenge to keep both online and offline content up to date.

But while the basic material is rather resistant to change, the internet setting allows mentors to easily provide and exchange additional material through the learning management system (or on their own homepages), with the entire information basis of the Internet at hand. In addition, the nonlinear material presentation simplifies individual adaptations to the course, such as skipping chapters or changing the chapter order.

4.2 Internet-based Communication

While online communication is part of any e-learning situation, integrating communication into the training portion of a course requires a sophisticated and stable communication environment, and discipline on the mentor's side to enforce the use of the chosen communication media.

Insisting on the use of a course specific discussion board for certain exercises proved helpful for both sides. Students are shy about bulletin boards because the "stupid question" that one has asked seems to be semi-permanent. But after some experience, they begin to see the value of this kind of asynchronous public interaction. In addition it is an excellent opportunity for the mentor to find out about uncertainties and misconceptions and to explicitly discuss them.

As we explained above, a text chat is quite unsuitable for Software Engineering team interaction. We can "make-do" with using tabbed browsers and setting up link lists, as many text-based chat systems do not permit copy and paste of what has been typed, making it difficult to send a URL to the students to be discussed (see (17) for more on the didactics of chatting). We have started to use the Netucate system (8), which provides an audio and a text chat, as well as application sharing, and a whiteboard setting that is not too unstable. In this manner, the student presenting her work can have her desktop sent to all of the participants. While she is discussing a drawing, she can move her mouse to point to parts of the drawing. It is also possible for other students to join in and write or draw using the application being discussed, if the application permits this.

With the present state of the art, the more sophisticated a communication tool is, the more frustrating its setup may become. Microphones need calibrating, screen resolutions have to be adapted, etc. Often, one or two students will be doomed to be "quiet listeners", while next time everything may work well. Finding a way of overcoming this type of frustration is in a way part of the exercise. After some familiarization, students become quite proficient in online interaction - and have acquired a teamwork skill that is not always a part of traditional SE instruction.

4.3 The Mentor's Role

One thing we did not anticipate was that with the given non-linear teaching material, mentors need to be separately informed about the didactical concepts behind the material and exercise organization. For example, it was neither obvious that 16 units meant 16 weeks, nor that the exercises and the face-to-face classroom sessions are coordinated, nor that the use of different virtual discussion media is a vital part of the course.

Just offering the possibility of bulletin boards and chats was not enough - we had to add secondary motivation by grading the exercises according to their timeliness, thus enforcing the scheduled milestones, to make students use the offered support structures. Mentors need to understand that they are grading timeliness rather than quality, because the quality level is to be raised through team discussions, and the major intention of grading exercises is motivation and synchronization.

It is an interesting finding that the restricted time available for mentor-student interaction means that the role of the mentor is not the teacher of the material, but rather a referee, a source of impulses and corrections - and a guide through the maze of approaches, views, methods, tools, and vocabulary in the field.

After four semester's experience, we feel that this way of teaching SE is definitely effective, but it requires a lot of effort on the mentor's side. Virtual communication, even if supported by state-of-the-art multimedia techniques, is less efficient than direct face-to-face

communication. For example, a chat with more than 5-7 people is very hard to perform effectively, so chat groups have to be split if they exceed this number. Hence a small group of students requires a lot of mentoring time. On the other hand, you save the lecturing effort - and you hardly "lose" anyone on the way, although we do not yet have hard data to support this subjective impression.

5. Conclusions

Is online instruction adequate for university level instruction in Software Engineering? Most certainly yes! It is not the best of environments - a block course over two weeks with just 4-6 students would be an ideal situation, in our opinion. But it is definitely not just a "poor sister" of traditional face-to-face instruction.

The possibilities for hypermedial presentation of the vast and interrelated material on Software Engineering are challenging. Very often, dynamic visualizations provide unrivaled explanations for highly abstract procedures, such as stepwise refinement. Being restricted to virtual communication, the students learn to effectively perform teamwork, coordination, and communication using state-of-the-art internet-based techniques.

Despite all the extra work involved in virtual vs. traditional teaching, we must say that we quite enjoy it!

6. References

- (1) Arnold, P.; Kilian, L.; Thillosen, A.: Pädagogische Metadaten im e-Learning: allgemeine Problemfelder und exemplarische Fragestellungen am Beispiel der Virtuellen Fachhochschule. In: Digitaler Campus. Vom Medienprojekt zum nachhaltigen Medieneinsatz in der Hochschule. (Proceedings of the GMW 2003 conference). Waxmann-Verlag 2003
- (2) Crutzen, C.K.M., 'Questioning gender in e-learning'. In: Grenzgänge; Genderforschung in Informatik und Naturwissenschaften, S. Schmitz and B. Schinzel (eds.), Ulrike Helmer Verlag, 2004
- (3) Görlitz, G.; Grimm, O., Müller, S., 2003, "Soziale Kompetenzen in der Online-Programmierausbildung," Informatik 2003 - Innovative Informatikanwendungen, Band 2, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V., K. Dittrich, W. König, A. Oberweis, K. Rannenberg, W. Wahlster (Eds.) Bonn, pp. 50-55.
- (4) Görlitz, G.; Grimm, O., Müller, S.; Schrade, E., 2003, "Design und Produktion von Online-Lernmaterial," e-Learning and beyond, Proceedings of the Workshop on e-Learning 2003, HTWK Leipzig 14-15. Juli 2003, pp. 131-138.
- (5) Görlitz, G.; Müller, S., 2003, "The coordinated process for the design and production of interactive e-learning content," Book of Abstracts, Online Educa Berlin, ICWE, Berlin, ISBN 3-9808909-2-9, pp. 138-140

- (6) Koper, R., 2003, Combining re-Usable Learning Resources to Pedagogical Purposeful Units of Learning In: Littlejohn, A. and Buckingham Shum, S. (Eds.) Reusing Online Resources (Special Issue) Journal of Interactive Media in Education, ISSN:1365-893X. <http://www-jime.open.ac.uk/2003/1/>
Commentary by Dai Griffiths and Rocio Garcia
- (7) Leinonen, T. 2005, Learning objects - Is the King naked? Free, Libre and Open Source Software in Education, <http://flosse.dicole.org/?item=learning-objects-is-the-king-naked> , May 19, 2005.
- (8) Netucate: <http://www.netucate.de>
- (9) Neue Medien in der Bildung: <http://www.medien-bildung.net>
- (10) Nielson, J., 2003, "PDF: Unfit for Human Consumption," Alertbox from July 14, 2003, <http://www.useit.com/alertbox/20030714.html> .
- (11) Schulmeister, R., 2003, "Taxonomy of Multimedia Component Interactivity. A Contribution to the Current Metadata Debate,". Studies in Communication Sciences. Studi di scienze della comunicazione. Special Issue, pp. 61-80. Also available online at <http://www.izhd.uni-hamburg.de/pdfs/Interactivity.pdf> .
- (12) Schulmeister, R., 1996, "Grundlagen hypermedialer Lernsysteme. Theorie - Didaktik - Design." Addison-Wesley: Bonn, Paris u.a.
- (13) Sosteric, M. and Hesemeier, S., "When is a Learning Object not an Object: A first step towards a theory of learning objects", In: International Review of Research in Open and Distance Learning (October - 2002) ISSN: 1492-3831. <http://www.irrodl.org/content/v3.2/soc-hes.html>
- (14) Stanierowski, M., n.d. "Softwaretechnik. Lehrbriefe 1-5," Hamburger Fern-Hochschule.
- (15) Virtuelle Fachhochschule: <http://www.oncampus.de>
- (16) Weber-Wulff, D., 2005, Pseudochat Software-Engineering, <http://www.f4.fhtw-berlin.de/~weberwu/eleed/LearnSE/chat01.pdf>
- (17) Weber-Wulff, D., 2003, "Teaching by Chat," Digitaler Campus. Vom Medienprojekt zum nachhaltigen Medieneinsatz in der Hochschule. M. Kerres, B. Voß (Eds.), Vol. 24, Medien in der Wissenschaft, Waxmann Verlag, pp. 366-375.
- (18) Weber-Wulff, D.; Kania, J.-P., 2002, "Interactions in On-line Learning," Open IFIP-GI-Conference on Social, Ethical and Cognitive Issues of Informatics and ICT, (SEC III), July a22-26, 2002, Dortmund, <http://vfh.fth-berlin.de/public/oldvfh/papers/interaction/Interactionsinonlinelearning.html>
- (19) Weber-Wulff, D., 1995, "Teambildung in Programmierung und Software-Engineering Kursen," Software Engineering im Unterricht der Hochschulen (SEUH '95), Bremen, A. Spillner, U. Breyman (Eds.), Teubner, Stuttgart, pp. 82-89. Available online at <http://www.f4.fhtw-berlin.de/~weberwu/papers/team.ps>
-

- [1] European Credit Transfer System, 1 credit is approx. 30 hours of work.
- [2] Herczeg et al, Universität Lübeck, <http://www.imis.uni-luebeck.de/de/forschung/projekte/vfh.html>
- [3] WikiWikiWeb: <http://c2.com/cgi/wiki?WikiWikiWeb>
- [4] <http://musoft.cs.uni-dortmund.de:8080/musoft/auto?expr=LearningObject>
- [5] Hamburger Fern-Hochschule, Alter Teichweg 19, D-22081 Hamburg , <http://www.hamburger-fh.de>
- [6] One of the good collaborative editing systems, the SubEthaEdit tool, is unfortunately only available for the Macintosh operating system (<http://www.codingmonkeys.de/subethaedit/>).
- [7] A streaming version of the film is available at <mms://edumedia.f1.fhtw-berlin.de/evideo/audiochat.wmv>
- There are also two versions available for download:
eeled.campussource.de/_/download/206/audiochat.avi (High Quality 550MB)
eeled.campussource.de/_/download/206/audiochat-cif-divx.avi (DivX Codec 128MB)
- The script is available at
eeled.campussource.de/_/download/206/DrehbuchAudiochat.doc ,
but bears little resemblance to the final film, which was adapted on short notice to fit the only two students willing to participate.

Credits:

Script: Ilse Schmiedecke

Mentor: Ilse Schmiedecke

Students: Gabriele Wohnsdorf and David Zellhöfer

Camera and Editing: Sebastian Hensel from the FHTW-Projekt eVideo (<http://evideo.fhtw-berlin.de>)

Producer: Debora Weber-Wulff

Copyright 2005 by the FHTW Berlin, all rights reserved (videos).

Please contact the authors of this paper if you wish to use the videos.